9-1999

QoS Scalability for Streamed Media Delivery

Charles Krasic
Oregon Graduate Institute of Science & Technology

Jonathan Walpole
Oregon Graduate Institute of Science & Technology

Citation Details

"QoS Scalability for Streamed Media Delivery," Charles Krasic and Jonathan Walpole, Oregon Graduate Institute School of Science & Engineering Technical Report CSE-99-011, September, 1999.

QoS Scalability for Streamed Media Delivery

Charles Krasic Jonathan Walpole

Oregon Graduate Insitute of Science & Technology
Department of Computer Science and Engineering

P.O. Box 91000, Portland, OR 97291-1000, USA
email: {krasic,walpole}@cse.ogi.edu

Abstract

Applications with real-rate progress requirements, such as media-streaming systems, are difficult to deploy in shared heterogenous environments such as the Internet. On the Internet, media-streaming systems must be capable of trading off resource requirements against the quality of the media streams they deliver, in order to match wide-ranging dynamic variations in bandwidth between servers and clients. Since quality requirements tend to be user- and task-specific, mechanisms for capturing quality of service requirements and mapping them to appropriate resource-level adaptation policies are required. In this paper, we describe a general approach for automatically mapping user-level quality of service specifications onto resource consumption scaling policies. Quality of service specifications are given through utility functions, and priority packet dropping for layered media streams is the resource scaling technique. The approach emphasizes simple mechanisms, yet facilitates fine-grained policy-driven adaptation over a wide-range of bandwidth levels. We demonstrate the approach in a streaming-video player that supports user-tailorable quality adaptation policies both for matching its resource consumption requirements to the capabilities of heterogeneous clients, and for responding to dynamic variations in system and network load.

Keywords: QoS — streaming — Internet — adaptive

1 Introduction

The Internet has become the default platform for distributed multimedia, but the computing environment provided by the Internet is problematic for streamed-media applications. Most of the well-known challenges for streamed-media in the Internet environment are consequences of two of its basic characteristics: end-point heterogeneity and best-effort service. The end-point heterogneity characteristic leads to two requirements for an effective streamed-media delivery system. First, it must cope with the wide-ranging resource capabilities that result from the large variety of devices with access to the Internet and the many means by which they are connected. Second, it must be able to tailor quality adaptations to accomodate diverse quality perferences that are often task- and user-specific. A third requirement, due to best-effort service, is that streamed-media delivery should be able to handle frequent load variations. Much of the research in the field of quality of service (QoS) is now concerned with addressing these requirements in the design of distributed multimedia systems.

The term *QoS* is often used to describe both presentation level quality attributes, such as the frame-rate of a video, and resource-level quality attributes, such as the network bandwidth. To avoid potential confusion in this paper, we distinguish *presentation QoS* from *resource QoS*. We use the term *QoS scalability* to mean the capability of a streamed-media system to dynamically trade-off presentation-QoS against resource-QoS.

The simplest approach to QoS scalability, used by many popular streamed-media applications, is to provide streamed-media at multiple *canned* quality levels. In this approach, end-host heterogeneity is addressed in the sense that a range of resource capabilities can be covered by the set of canned levels, but it fixes the choice of quality adaptation policy. Furthermore, dynamic load variations are left to be dealt with by the client-side buffering mechanism. Normally a buffering mechanism is associated with concealment of jitter in network latency, but it can also be used to conceal short-term bandwidth variations—if the chosen quality level corresponds to a bandwidth level at, or below, the average available bandwidth. In practice, this approach is too rigid. Client-side buffering is unable to conceal long-term variations in available bandwidth, which leads to service interruptions when buffers are overwhelmed. From a user's perspective, interruptions have a very high impact on the utility of a presentation. To avoid interruption, the user must subscribe to quality levels that drastically under-utilize their typical resource capabilities. The canned approach is also difficult from the provider's perspective. Choosing which canned levels to support poses a problem because it is difficult for a provider to know in advance how to best partition their service capacities. The canned approach fails to deal with best-effort service or heterogeneity.

Recently, in search of Internet compatible solutions, researchers have begun to explore more-adaptive QoS-scalability approaches. There are two classes of such approaches. The first class, data rate shaping (DRS), performs some or all of the media encoding dynamically so that the target output rate of the encoder can be matched to both the end-host capabilities and the dynamic load characteristics of the network [5]. The other class of approaches is based on layered transmission (LT), where media encodings are split into progressive layers and sent across multiple transmission channels [8, 11]. The advantage of DRS is that it allows fine-grained QoS scalability, that is, it can adjust compression level to closely match the maximum available bandwidth. Since LT binds layers to transmission channels, it can only support coarse-grain QoS scalability. On the other hand, LT has advantages stemming from the fact that it decouples scaling from media-encoding. In LT, QoS scaling amounts to adding or removing channels, which is simple, and can be implemented in the network through existing mechanisms such as IP multicast. In stored-media applications, LT can perform the layering offline, greatly reducing the burden on media servers of supporting adaptive QoS-scalability.

A universal problem for QoS scalability techniques arises from the multi-dimensional nature of presentation-QoS. QoS dimensions for video presentations include spatial resolution, temporal resolu-

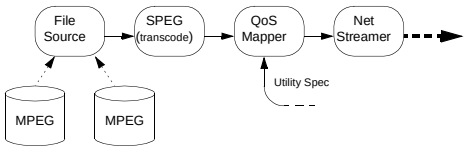


Figure 1: Server Side Pipeline

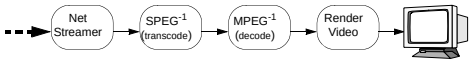


Figure 2: Client Side Pipeline

tion, color fidelity, etc. Yet QoS scalability mechanisms such as DRS and LT expose only a single adaptation dimension, output rate in the case of DRS, or number of channels in the case of LT. The problem is mapping multi-dimensional presentation-QoS requirements into the single resource-QoS dimension. In both LT and DRS, the approach has been to either limit presentation-QoS adaptation to one dimension, or to map a small number of presentation-QoS dimensions into resource QoS with ad-hoc mechanisms. DRS and LT provide only very primitive means for specification of QoS preferences.

In this paper we present a new model for adaptive QoS scalability, based on two mechanisms: *utility functions*, for capturing presentation QoS requirements; and *priority packet-dropping* (PPD) with layered media, a technique for resource-QoS scaling. Utility functions are a general *declarative* means for specification of user- and task-specific presentation-QoS preferences. In PPD, an alternative to DRS and LT, layered media streams are split into packets in such a way that data corresponding to distinct presentation QoS dimensions are assigned to distinct packets. PPD offers a middle ground between DRS and LT. On the one hand, like DRS, QoS scaling can be fine grained, since the unit of adaptation in PPD is the packet. On the other, like LT, encoding and scaling actions are decoupled because PPD doesn't require a full encoder at delivery time. Furthermore, we show that PPD facilitates mapping multi-dimensional presentation-QoS preferences into resource-QoS adaptation plans encoded in packet priorities; that is, we describe how to derive the appropriate packet-priority labeling from utility functions.

We demonstrate this approach to adaptive QoS scalability in an implementation of a streamed video player. We characterize the player's performance by showing the measured relationships between user requirements, presentation quality, and resource consumption. Our characterization shows that together, utility functions and the PPD mechanism lend themselves well to supporting *user-tailorable* adaptation in *multiple quality dimensions*. We also show that at the resource level, adaptation can be done at a *fine granularity*.

The rest of this paper is organized as follows. Section 2 describes our model of streamed-media delivery, and outlines the organization of our player. Section 3 explains how we use utility functions to capture presentation QoS requirements. Section 4 describes the SPEG extension, which adds SNR-scalability layers to MPEG-1 video. Section 5 explains the QoS Mapper algorithm. Section 6 presents our experimental results. Section 7 discusses related work; and section 8 concludes.

2 Media Streaming with Priority Packet Dropping

Our model of streamed-media delivery is based on pipelines composed of producer-consumer elements. The player we have implemented has a video pipeline consisting of a server side and client side as illustrated in figures 1 and 2. The pipeline contains the following elements:

- A pair of transcoders, to convert between standard MPEG-1 video and our own layered format, SPEG.
- A QoS mapper. This element translates a QoS policy into a packet priority labelling for the video stream.
- An assortment of standard elements: a file source, a standard MPEG video decoder, and an uncompressed video renderer.

Media streams flow through the pipeline as a sequence of priority-labelled packets, and priority-based packet dropping is the exclusive means of quality adaptation. Normally, the effects of packet dropping on presentation QoS of continuous media streams are non-uniform, and can quickly result in an unacceptable presentation. For example, dropping the MPEG header bits of a picture typically renders the whole picture, and possibly a large segment of surrounding video, unusable[1]. In contrast, dropping several pixel blocks may be almost imperceptible to a user. Critical data such as the picture header bits are infrequent, but the consequences of dropping them are so high that random dropping quickly degrades playout. Our experience has been that random packet drop rates as low as 5–10% cause the MPEG video to become unusable.

In contrast to random dropping, informed packet dropping can achieve significant reductions in bandwidth while still delivering statisfactory presentation quality. Strategic elimination of entire pictures is the simplest and most common example of informed packet dropping for video[2, 13]. However, there are other ways to trade-off presentation QoS to reduce resource QoS requirements, such as adapting the spatial resolution and the color fidelity[14]. The correct adaptation policy is often task- or user-specific. For example, a user viewing streamed video on on a small portable device such as a PDA might prefer preservation of temporal resolution, since spatial resolution is limited on their relatively small screen anyway, whereas a desktop user receiving the same video might give a higher preference to spatial detail.

The problem with supporting a tailorable adaptation policy is that it leads to complex control problems in streamed media systems [2, 13, 10, 7]. To control the complexity of supporting tailorable adaptation, we chose to explore the simple mechanisms of utility functions and priority packet dropping.

We describe in section 4 how video data of orthogonal presentation QoS dimensions are partitioned into packets in our SPEG video format. In sections 3 and 5 we describe quality specification through utility functions and the algorithm used by the QoS mapper to translate utility functions into priorities.

3 QoS Specification

A utility function is a simple and general means for users to specify their preferences. Figure 3 depicts the general form of a utility function. The horizontal axis describes an objective measure of lost quality, while the vertical axis describes the subjective utility of a presentation at each quality level. The region between the q_{max} and q_{min} thresholds is where a presentation is acceptable. The q_{max} threshold marks the point where lost quality is so small that

[1] In an MPEG movie using the common 15 picture group of pictures (GOP) pattern, a missed I-Frame header would yield 0.5 second interruption.

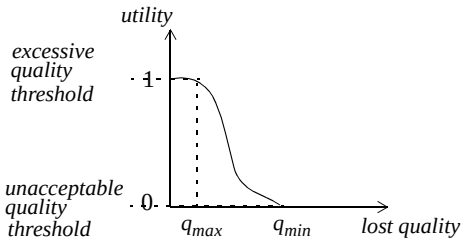


Figure 3: A utility function with thresholds

the user considers the presentation "as good as perfect." The area to the left of this threshold, even if technically feasible, brings no additional value to the user. The rightmost threshold q_{min} demarks the point where lost quality has exceeded what the user can tolerate, and the presentation is no longer of any use. The utility levels on the vertical axis are normalized so that zero and one correspond to the "useless" and "as good as perfect" thresholds. In the acceptable region of the presentation, the utility function should be continuous and monotonically decreasing, reflecting the notion that decreased quality should correspond to decreased utility.

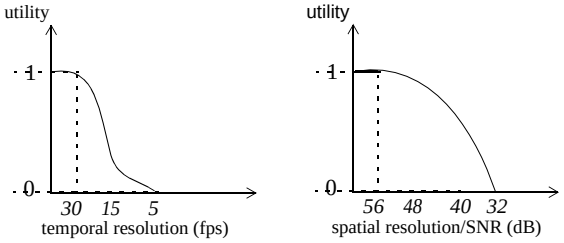


Figure 4: Sample Utility Functions: Temporal and Spatial Resolution. Temporal utility has its q_{max} threshold at 30 frames per second (fps), which corresponds to zero loss for a typical digital video encoding. The q_{min} threshold, beyond which the presentation is considered unusable, for temporal resolution corresponds to 5 fps. Spatial resolution is expressed in terms of the Signal to Noise Ratio (SNR). SNR is a commonly used measurement for objectively rating image quality.

Utility functions are declarative, they do not specify anything directly about how to deliver a presentation, in particular they do not require any knowlege of resource-QoS trade-offs from the user. Furthermore, they represent the adaptation space in an idealized continous form, even though QoS scalability mechanisms can often only make discrete adjustments in quality. By using utility functions to capture user preferences, this declartive approach avoids commitment to resource QoS and low-level adaptation decisions, leaving more flexibility to deal with the heteregeneity and load-variations of a best-effort environment. Figure 4 gives a pair of example utility functions for video. In our model, a utility function can be specified for each presentation-QoS dimension over which the system allows control.

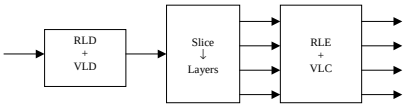


Figure 5: MPEG to SPEG transcoding. The MPEG stream is partially decoded, undoing the run-length (RLD) and variable-length huffman (VLD) codings. Slices, sequences of pixel blocks, are partitioned into layers, and then run-length (RLE) and huffman (VLC) coding are re-applied

4 SPEG: A simple extension to MPEG-1

In order to experiment with mapping multi-dimensional QoS requirements into priority labeling for layered media, we needed a suitable layered media codec. Although layered scalability extensions are present in common standards such as MPEG-2, freely available implementations are not available. For our purposes, the easiest solution was to develop a rapid prototype with the commonly available MPEG-1 codec [9].

A scalable encoding allows the decode process to trade-off the amount of work performed against the resulting quality of the decoded data. Scalable encodings often take a layered approach, where the data in an encoded stream is divided conceptually into layers. A base layer can be decoded into presentation form with a minimum level of quality. Extended layers are progressively stacked above the base layer, each corresponding to a higher level of quality in the decoded data. An extended layer requires lower layers to be decoded to presentation form.

4.1 Adding scalability to MPEG

One of the key parameters governing the compression rate in MPEG encoders is the quantization level, that is the number of bits dropped from the coefficients of the frequency domain representation of the image data. The degree to which an MPEG video encoder can quantize is governed by the trade-off between the desired amount of compression and the final video quality. Too much quantization leads to visible video artifacts. In standard MPEG-1 video, the quantization levels are fixed at encode time. In SPEG, we layer the video by iteratively increasing the quantization by one bit per layer. At run time, quantization level may be adjusted by selection of some or all of the layers.

Rather than constructing an entirely new encoder, our approach is to transcode MPEG-1 video into the SPEG layering. Transcoding has lower compression performance than a native approach, but is easier to implement than developing a new scalable encoder. It also has the benefit of being able to easily use existing MPEG videos. For stored media, the transcoding is done offline.

Figure 5 describes the transcoder organization. Transcoding starts by partially decoding the original MPEG-1 video. This process involves parsing video headers, then applying inverse entropy coding (VLD + RLD) to produce slices[2]. Data from slices is partitioned into layers and then re-compressed with a forward entropy code (RLE + VLC).

Figure 6 depicts how SPEG partitions data from MPEG blocks among SPEG layers. Consider n layers in SPEG, the base layer is numbered 0 and the extension layers are numbered 1 to $n-1$. A DCT block in the highest extension layer is coded as the difference between the original MPEG DCT block, and the original block with one bit of precision removed. Generalizing this approach, SPEG

[2]In MPEG video, slices are sequences of frequency-domain (DCT) coefficient blocks

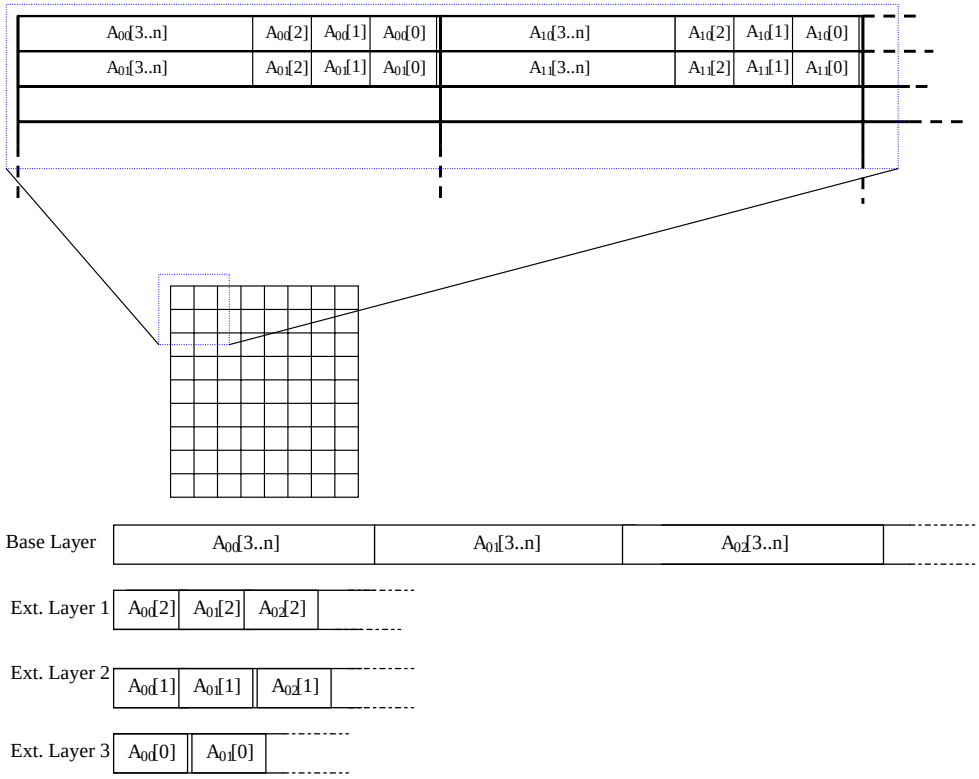


Figure 6: SPEG spatial resolution layers. MPEG transforms 8x8 blocks of pixels into 8x8 blocks of coefficients from the 2-dimensional discrete-cosine transform (DCT). Each coefficient is denoted A_{ij}, and the bits of an individual coefficient are indexed $A_{ij}[0..n]$. Quantization, dropping low-order bits, is the main lossy component of the MPEG compression technique, resulting in loss of spatial detail. In SPEG transcoding, the low-order bits of each MPEG coefficient are progressively assigned to layers starting from the highest extension layer and ending with the lowest, with the base layer containing all remaining bits. Eliminating SPEG layers corresponds to increasing quantization in the MPEG encoding process.

codes extension layers as the difference between the original block with k bits removed and $k-1$ bits removed, where $n-k$ equals the extension layer number. The base layer is coded as the original block with $n-1$ bits removed. Notice that extension layers are differences while base is not. Once layered, entropy coding is re-applied. Our current SPEG implementation uses four layers, the base plus three extension layers.

Partitioning of SPEG data occurs at the MPEG slice level. All header information from the original MPEG slice goes unchanged into the base layer slice, along with the base layer DCT blocks. Extension slices contain only the extension DCT block differentials.

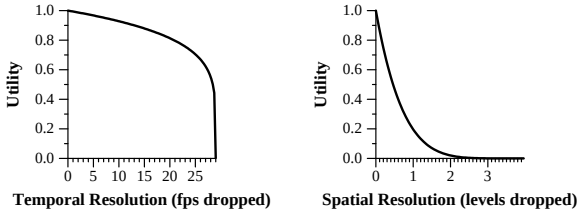
The SPEG to MPEG transcode that returns the video to standard MPEG format is performed online as part of the streamed-media pipeline, and consists of the same steps as the MPEG to SPEG transcoding, only in reverse. This approach meant that we had to make only minor changes to fit the public domain Berkeley `mpeg_play` decoder into our pipeline.

5 QoS Mapper

In this section, we describe our algorithm for translating presentation QoS requirements, in the form of utility functions, into priority assignments for packets of a media stream, such as SPEG. We call the pipeline element that executes this algorithm the QoS *mapper*. The key features of the mapper are the following: it is designed to run dynamically as part of the streamed-media delivery pipeline; it works on multiple QoS dimensions; and it does not require a priori knowlege of the presentation to be delivered.

The mapper assumes several characteristics of the media formats it processes. The first assumption is that data for orthogonal quality dimensions are in separate packets. The second assumption is that the presentation-QoS, in each available dimension, can be computed or approximated for subsequences of packets. Finally, the third assumption is that any media-specific packet dependencies are known by ost the mapper. Our player fragments an SPEG stream into packets in way that ensures these assumptions hold. The packet format we use for SPEG is based on the RTP format for MPEG video[4], with additional header bits to describe the SPEG spatial resolution layer of each packet. This approach is an instance of application-level framing [3]. Our format requires that each packet contain data for exactly one SPEG layer, enforcing the first assumption of the mapper holds. Further, the packet header bits convey sufficient information to compute presentation QoS of sequences of packets and to describe inter-packet dependencies, satisfying the second and third assumptions. Since all the information needed by the mapper is contained in packet headers, the mapping algorithm does not need to do any parsing or processing on the raw data of the video stream, which is very important in limiting the computational cost of mapping. With these assumptions in mind, we can now describe the mapping algorithm

The mapping algorithm computes a priority for each packet as follows. The mapper starts by analyzing the packet stream headers, and computing for each packet the lost presentation QoS that would result if the packet were dropped. The lost presentation-QoS calculation is done for each QoS dimension. The mapper then converts, for each QoS dimension, lost presentation QoS into lost utility by using the user-provided utility functions. The maximum of the per-dimension lost utilties for a packet is used as the packet's overall priority. We describe these steps in greater detail next, with an example based on the SPEG implementation.



(a) Temporal Utility (b) Spatial Utility

Figure 7: Mapping Example: QoS Requirements

5.1 QoS Mapper: An SPEG example

Figure 7 presents utility functions for temporal and spatial resolutions of SPEG video. We have applied non-even bias to the utility functions to give spatial resolution more importance than temporal resolution, as is apparant in the differing slopes of the two lines.

Now we consider an example input SPEG movie that contains the following group of pictures (GoP) pattern:

$$I_0 B_1 B_2 B_3 P_4 B_5 B_6 B_7 \ldots.$$

The letter denotes the MPEG frame type, and the subscript is the frame number. For this example, we assume the SPEG packet sequence contains four packets for each frame, one for each of the four SNR layers supported by SPEG. The top-level of the mapper calls subroutines that compute the lost presentation QoS for each packet in the sequence, in each QoS dimension.

For the temporal resolution dimension, the lost QoS subroutine groups packets by frame and works by assigning a frame drop ordering to the sequence of frames. This algorithm uses a simple heuristic to pick an order of frames that minimizes the jitter effects of dropped frames. The ordering heuristic is aware of the frame dependency rules of SPEG. For example, the ordering always ensures that a B (bidirectional) frame is dropped before the I or P frames that it depends on. In our example packet sequence, the drop ordering chosen by the heuristic is:

$$B_1 B_5 < B_3 B_7 < B_2 B_6 < P_4 < I_0$$

where $<$ denotes the *dropped-before* relationship. With this ordering, the frame rate of each packet is computed according to its frame's position in the ordering. The packets of frame B_1 are assigned a reduced frame-rate value of $1/8 * 30$, since frame B_1 is the first frame dropped, and a frame rate of $30fps$ is assumed. Frame P_4 is assigned a reduced frame rate value of $7/8 * 30$ since it is the second last frame dropped. Notice that the lost QoS value is cumulative—it counts lost QoS from dropping the packet under consideration, plus all the packets dropped earlier in the ordering. These cumulative lost-QoS values are in the same units as the utility function's horizontal axis.

For the spatial resolution dimension, the lost QoS calculation is similar. Rather than computing ordering among frames, packets are grouped first by SNR level, and then sub-ordered by an even-spacing heuristic similar to the one used for temporal resolution. As a simplification, we approximate the spatial QoS loss for each packet by a function based on the average number of SNR levels, rather than the actual SNR value, present in each frame when the packet is dropped.

The mapper applies the utility functions from the user's quality specification to convert lost-QoS values of packets into cumulative lost-utility values. The final step is to combine the lost-utilities in the individual dimensions into an overall lost-utility that is the basis for the packet's priority. We assign the priority as follows: If

in *all* quality dimensions the cumulative lost utility is zero, assign minimum priority. If in *any* quality dimension the cumulative lost utility is one, assign maximum priority. Otherwise, scale the maximum of the cumulative lost dimensional utilities into a priority in the range [minimum priority + 1, maximum priority - 1]. Minimum priority is reserved for packets that should never pass, because the cumulative lost utility of the packet does not cause quality to fall below the q_{max} threshold. Hence the quality level does not enter the excessive region of the utility function. Similarly, the maximum priority is reserved for packets that should always pass since in at least one of the quality dimensions, dropping the packet would cause quality to drop below the q_{min} threshold. So in one or more dimensions, dropping the packet would cause the presentation to become useless.

Frame	*Priorities*			
I_0	10	9	7	5
B_1	9	7	5	3
B_2	9	7	5	3
B_3	9	7	5	3
P_4	10	8	6	4
B_5	9	7	5	3
B_6	9	7	5	3
B_7	9	7	5	3
I_8	15	11	9	6
...				

Figure 8: Mapping Example: Priority Assignment. Each row describes a single frame of the SPEG video. I_0 denotes frame number 0, which has an I type (intra coded). The columns describe the priority labelling for each spatial resolution layer within the frame, with base-layer leftmost.

Figure 8 shows a fragment of a the priority assignment made by the mapper for *Jackie Chan*, one of the movies used in our experiments in section 6. This assignment was made using the utility functions of figure 7, although the mapper considered more than eight frames when it made the assignment. In section 6 we show how this priority assignment translates to a quality adaptation policy, and discuss its resource implications (see figure 10).

5.2 Performance Considerations

If both the users's quality preferences and the content of media streams were known a priori, it would be possible to derive the optimal packet labelling. Since this information is only available dynamically, and then only a portion of the media content is known, we approximate the optimal labelling by considering a finite context, i.e. number of packets, from the packet sequence. As the amount of context considered increases, computational costs increase and the approximation of the optimal labelling improves. We call the number of packets considered by the priority assignment algorithm the priority window size. A related concern is the finite number of bits allocated to store priorities in packet headers, that is the number of available priorities, as it places another limit on the accuracy of priority assignment. The two are related in that the number of available priorities are only usable if the window has at least as many distinct packets as there are priorities; conversely, if the window contains many more packets than priorities, the optimality of the approximation is no better than with a smaller window. Stated another way, to truly realize an optimal labelling would require a window big enough to hold all of the packets of a presentation, and enough priority bits to label each packet uniquely.

For the experiments of the next section, our mapper implementation classifies packets by frame number and resolution level, so

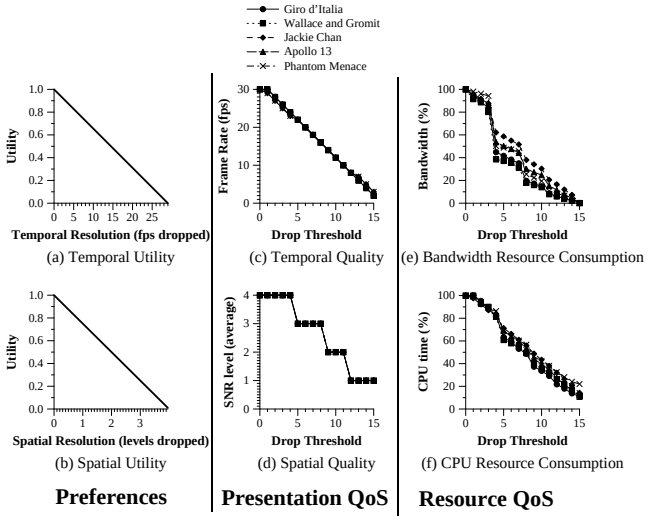


Figure 10: Adaptation Characterization for a Mixed Drop Policy

the priority window is expressed in units of frames rather than packets. We used a window size of 32 frames. Since each frame has 4 spatial resolution levels, that gives 128 distinct classes of packet per window. The priority field is 4 bits, giving 16 priorities, and so our chosen window size corresponds to 8 distinct classes of packet per priority level.

6 Experiments

In this section, we present results of experiments to characterize the adaptation performance of our approach. The adaptation performance is measured with respect to both presentation QoS and resource QoS. The experiments were conducted for three adaptation policies, as specified by different sets of utility functions. The presentation QoS results show that our approach supports tailorable adaptation in multiple QoS dimensions. The resource QoS results demonstrate that the adaptation covers a wide range and is fine grained.

The following sections are organized according to the three adaptation policies: a policy giving equal preference to spatial and temporal resolution is used in section 6.1, then a policy with extreme preference for temporal resolution in section 6.2, and finally extreme preference for spatial resolution in 6.3. For each of the three policies, two sets of results are given, one for presentation QoS and another for resource QoS.

The presentation-QoS adaptation profile shows how effective the QoS mapper algorithm is at assigning priority labeling that corresponds to the quality preferences specified by the utility functions. The resource-QoS adaptation profile illustrates, for a given priority labeling, how priority-drop threshold adjustments affect two basic resource requirements: network bandwidth, and client-side CPU. The profile data were gathered from instrumented runs of our player on a set of test videos. Figure 9 describes the set of videos we used, and the basic encoding parameters. The videos were chosen for their diversity so we could evaluate the sensitivity of our approach to different encoders and encoder parameters

6.1 An Evenly Balanced QoS Policy

In figures 10(a) and (b) we see simple linear utility functions for temporal and spatial utility where the relative QoS preferences are

Video	Resolution	Length (frames)	GOP pattern
Giro d'Italia	352x240	1260	BBIBBPBBPBBPBBP
Wallice and Grommit	240x176	756	IPI
Jackie Chan	720x480	2437	IBBBPBBB
Apollo 13	720x480	864	BBIBBP
Phantom Menace	352x240	4416	BIBPBPBPBPBPBPBP

Figure 9: Movie Inputs. The movies were coded with several different MPEG encoders. A variety of content types, movie resolutions, and GOP patterns were chosen to verify our techniques perform consistently.

equal across the respective ranges of adaptation. In this policy, and the others that follow, the utility function for temporal resolution is based on a 30 fps maximum rate, and a 1 fps minimum rate.

Figures 10(c) and (d), show the presentation QoS derived from this policy for various priority-drop thresholds. That is, given the priority-labeling produced by the QoS mapper, the graph shows what quality level is realized at each priority-drop threshold. Our current system implements sixteen priority levels. The priority drop thresholds are on the horizontal axes of the graphs. An increased priority drop threshold means more packets are dropped.

Ideally, the presentation-QoS graphs would look the same as the utility functions they were derived from. In particular, the range of acceptable presentation QoS would be covered, and the shape of adaptation would follow the shapes of the utility functions. Figure 10(c) shows the relationship between presentation-QoS for temporal resolution (frame rate) and priority-drop threshold. It should be noted that figure 10(c) contains lines for each of the test movies, but they overlap very closely because the mapper is able to label packets to follow the utility function policy closely. Although desirable, this result was not entirely expected because MPEG's interframe dependencies constrain the order in which frames can be dropped, and some GOP patterns are particularly poorly suited to frame dropping. On the spatial resolution side, in figure 10(d), we note that our current mapper drops resolution levels uniformly across all frames, resulting in a stair-shaped graph, since there are only 4 SNR levels in SPEG. In as much as the SPEG format allows, the presentation-QoS matches the specified user preferences.

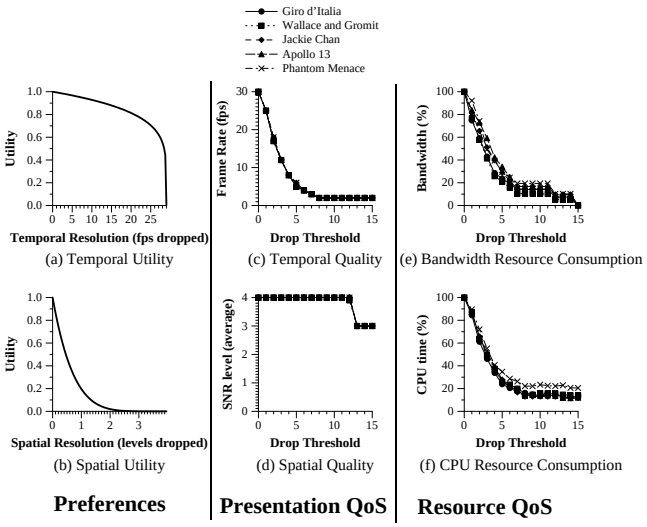


Figure 11: Adaptation Characterization for a Temporal Drop Policy

Resource QoS adaptation profiles are shown in the third pair of graphs in Figures 10(e) and (f). We show the average bandwidth of the movies at each drop threshold, as a percentage of the bandwidth when no packets are dropped. Similarly, we show the CPU time required for client side processing of the video (recall Figure 2) at each drop threshold. A good shape for these graphs would be smooth and linear over a wide range of resource levels. We see that bandwidth in Figure 10(e) does indeed range all the way down to only a few percent, although there is a rather sharp drop when the first SNR layer is dropped. CPU time in Figure 10(f) is very nice and smooth, although it does not cover as much range as bandwidth, and reaches a minimum of about 10 percent. We also note that the movies are closely clustered in their resource-QoS graphs, indicating that adaptation is independent from differences in encoders or encoder parameters.

6.2 A Temporal Drop Policy

Figures 11(a) and (b) show utility functions for temporal and spatial utility where the QoS dimensions have been biased to give an extreme preference to temporal resolution. We see in figure 11(c) that frame rate decreases over most of the adaptation space. Near the end of the adaptation space, the frame rate reaches the minimum allowed by the QoS utility function and spatial resolution is dropped (figure 11(c)). The final drop in spatial resolution does not drop to the minimum possible spatial resolution because of the uniform near zero utility in the corresponding range of figure 11(a).

The resource implications of this adaptation policy are shown in figures 11(e) and (f). Again, we see smooth and wide-ranging adaptation spaces for both dimensions. There is still a noticeable drop in bandwidth when the first SNR level is dropped, but it is less pronounced than in the mixed-drop policy.

6.3 A Spatial Drop Policy

Figure 12 summarizes the next experiment, where we reverse the preference between spatial and temporal resolution from the previous. The temporal utility function 12 (a) has the abrupt drop at the minimum-loss threshold (1/30), emphasizing that the presentation utility drops abruptly when temporal resolution is lost. The presentation QoS profiles in figures 12 (c) and (d) are similar to the Temporal Drop policy, except that the spatial Quality can not be adjusted in as smooth a manner as the frame rate. The interesting result here is in the resource profiles of figures 12 (e) and (f). The bandwidth reductions from increasing the drop threshold are more pronounced than the reductions in CPU. This reflects the fact that the MPEG decoder does a significant amount of work regardless of the resolution level of the image. An implication of this is that it might be better to allow more adaptation in the temporal resolution dimension when the CPU is a bottleneck resource. Since our architecture allows for dynamic replacement of QoS policy quite naturally, we can imagine implementing re-mapping according to resource-bottleneck demands. This possibility will be examined in future work.

Video	MPEG bandwidth (Mbps)	SPEG bandwidth (Mbps)	Increase bandwidth (%)	MPEG CPU (secs)	SPEG CPU (secs)	Increase CPU (%)
Giro d'Italia	1.823	2.121	16.3	45.8	72.9	59
Wallice and Grommit	0.968	1.081	12.7	12.1	16.6	37
Jackie Chan	1.839	2.479	34.8	216	252.2	17
Apollo 13	3.474	4.193	20.7	89.3	121.5	36
Phantom Menace	1.228	1.313	6.9	103.7	180.4	74

Figure 13: Overhead of SPEG

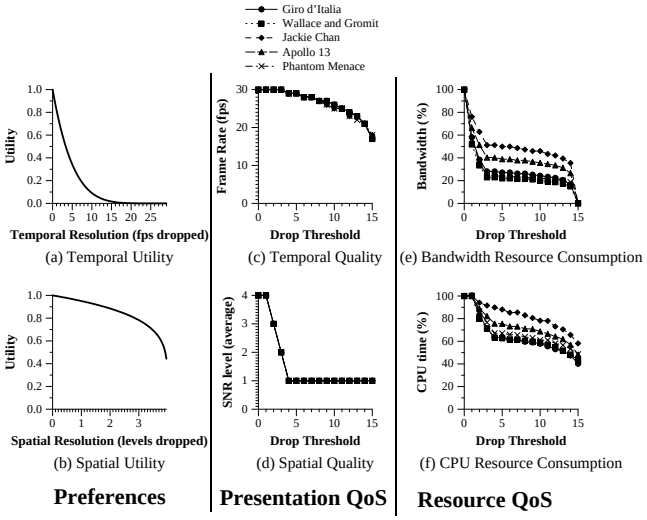


Figure 12: Adaptation Characterization for a Spatial Drop Policy

6.4 The Price of Adaptation

We now describe some of the performance costs associated with dynamic quality adjustment. Figure 13 compares performance for MPEG and SPEG versions of movies at the same presentation QoS level.

Given the wide range of adaptation shown in the adaptation experiments, the relatively small bandwidth overhead of SPEG is encouraging, especially considering the simplicity of the approach used in SPEG. The CPU overhead is more severe, but we know there is great room for improvement. The choice of transcoding SPEG back to MPEG was convenient for constructing the experiment, but is an obvious major source of un-necessary overhead.

7 Related Work

There are several approaches to data-rate shaping of MPEG video that relate to our SPEG extension. The basic approach we used to drop spatial resolution information from MPEG video, post-encode quantization, was taken from Yeadon's dissertation work and prototype code for QoS filters[14]. Yeadon proposes an architecture and signaling protocol for doing DRS in the network, for the purpose of addressing heterogeneity in multicast distribution trees. Our priority packet-drop mechanism requires much more modest support from the network, due to the decoupling of scaling (packet dropping) from encoding (layering). Jacobs and Eleftheriadis [5] performed more rigorous analyses of SNR scaling of MPEG video than Yeadon, in their work on Data Rate Shaping (DRS). Their analysis includes an optimality result for DRS, as well as a result showing that a much cheaper algorithm which avoids Huffman decoding, thus having greatly reduced complexity, achieves near-optimal results. Their work was focused on SNR scaling alone, and did not consider adapting multiple dimensions. As with Yeadon's approach, we could adapt Jacobs and Eleftheriadis' DRS to a layered packet encoding, most likely with a significant improvement over our current SPEG format. We were unaware of these results at the time of the initial SPEG implementation. However we believe the Yeadon derived approach delivered sufficient initial results. Zeng and Liu [15] introduce a novel variation on SNR scalability where rather than re-quantize MPEG, they drop entire blocks and then reconstruct the picture area using interpolation. Zeng and Liu show that their approach has better rate-distortion performance than DRS. On the other hand, their approach is likely to have higher CPU cost at the client-side, and the interpolation would require fairly major changes to the basic MPEG decoder. As with Jacobs and Eleftheriadis, they did not consider multi-dimensional scaling.

Receiver-driven Layered Multicast (RLM) [8] is a comprehensive solution for streamed delivery of media over IP multicast. In RLM, each transport layer is assigned to a distinct multicast group. Receivers adapt QoS by joining or leaving groups. The basic idea is that when receivers detect congestion, they drop groups, and when they detect available bandwidth, they join groups. Our work has many of the same overall objectives as RLM, although we start from one significantly different base assumption. We assume a packet-drop model versus uniform layer dropping. We also explore capture of user preferences and mapping in more detail than RLM. Recently, the notion of *TCP-friendliness* has become a very prominant research topic in networking. Part of our motivation in exploring priority packet drop is that it is easier to map to TCP's model of sender-side congestion control, as opposed to RLM's receiver driven congestion control. Although, Vicisano et al. have proposed a modification to RLM with the objective of TCP friendly congestion control [12], and Bajaj et al. have conducted an interesting preliminary, albeit inconclusive, comparison of priority dropping and layered transmission [1], it remains unclear whether RLM can be fine-grained enough or TCP friendly enough for wide scale adoption. We leave these issues to future work.

Utility functions for capturing QoS specifications have been studied by numerous groups. We note that our utility functions relate utility to *presentation* QoS, where most other QoS specifications are concerned with relating utility to *resource* QoS. In the QoS Resource Allocation Model (Q-RAM) by Rajkumar et al.[10], a model for resource allocation based on spaces of m resource dimensions and n quality dimensions is described. Application-level specifications are given as graphs between user utility and QoS, while architectural level specifications are graphs between QoS and resource levels. Q-RAM includes QoS mapping algorithms based on optimization heuristics, which are described for various dimensionalities (combinations of m and n) both when dimensions are independent and dependent. Our mapping algorithm is significantly different to Q-RAM's in that we take overall utility

to be the maximum of individual dimensional utilities, rather than a weighted sum. This approach simplifies the optimization problem greatly. Kravets et al. [6] describe how to deploy utility specifications, called payoff functions in their terminology, in layered distributed software architectures. They use payoff functions to guide trade-offs between communication reliability and communication latency. Since these payoff functions relate resource QoS to utility, rather than presentation QoS to utility as we do, their approach is suited for use at application design time rather than for capturing preferences for dynamic user- and task-specific requirements.

8 Conclusions

QoS Scalability will be essential to reconciling the Internet's characteristics of end-point heterogeneity and best-effort service with its vast potential as a platform for streamed-video delivery. We have described a QoS scalability approach with two main contributions: utility functions for capturing user preferences, and priority-packet drop delivery of layered media for resource-QoS scaling. Our experiments have confirmed that this approach to QoS scalability has the following major benefits: it allows sophisticated control by user-tailorable quality adaptation policies, it allows delivery over a wide range of resource levels, and within that range resource usage can be matched at a fine grain. In addition to showing the effectiveness of the approach, our experiments show that the overhead is low enough to perform online in software, making it feasible to deploy for Internet streaming.

9 Acknowledgements

This research was supported in part by DARPA contracts/grants N66001-97-C-85222, N66001-97-C-8523, and F19628-95-C-0193, and by Tektronix, Inc., and Intel Corporation.

We would like to thank Mark Jefferys for writing the SPEG implementation, and the rest of the members of the Quasar project who participated in the development of the video-pipeline software. Anne-Francoise LeMeur, Calton Pu, David Steere, and Dylan McNamee provided many valuable comments on drafts of this paper.

References

[1] Sandeep Bajaj, Lee Breslau, and Scott Shenker. Uniform versus priority dropping for layered video. In *Computer Communication Review*, Vancouver, B.C., October 1998.

[2] Shanwei Cen. *A Software Feedback Toolkit and its Application In Adaptive Multimedia Systems*. PhD thesis, OGI, October 1997.

[3] David D. Clark and David L. Tennenhouse. Architectural considerations for a new generation of protocols. In *Proceedings of SIGCOMM'90 Symposium Communications Architectures & Protocols*, Philidelphia, Pennsylvania, September 1990.

[4] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar. RTP Payload Format for MPEG1/MPEG2 Video. RFC 2250, January 1998.

[5] Stephen Jacobs and Alexandros Eleftheriadis. Streaming video using dynamic rate shaping and tcp flow control. *Visual Communication and Image Representation Journal*, January 1998. (invited paper).

[6] Robin Kravets, Ken Calvert, and Karsten Schwan. Payoff adaptation of communication for distributed interactive applications. *The Journal for High Speed Networking: Special Issua on Multimedia Networking*, Winter 1999.

[7] Baochun Li, Dongyan Xu, Klara Nahrstedt, and Jane W.-S. Liu. End-to-end QoS support for adaptive applications over the internet. In *Proceedings of SPIE Symposium on Voice, Video and Data Communications*, Boston, Massachusetts, November 1998.

[8] Steven McCanne, Martin Vetterli, and Van Jacobson. Low-complexity video coding for receiver-driven layered multicast. *IEEE Journal on Selected Areas in Communications*, 16(6):983–1001, August 1997.

[9] K. Patel, B. C. Smith, and L. A. Rowe. Performance of a software mpeg video decoder. In *Proceedings ACM Multimedia 93*, pages pp. 75–82, Anaheim, CA, August 1993.

[10] R. Rajkumar, C. Lee, J. Lehoczky, and D. Siewiorek. A resource allocation model for QoS management. In *In Proceedings of the IEEE Real-Time Systems Symposium*, December 1997.

[11] Dorgham Sisalem and Frank Emanuel. QoS control using adaptive layered data transmission. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, Austin, Texas, June 1998.

[12] Lorenzo Vicisano, Luigi Rizzo, and Jon Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proceedings of INFOCOM'98*, San Francisco, March 1998.

[13] Jonathan Walpole, Rainer Koster, Shanwei Cen, Crispin Cowan, David Maier, and Dylan McNamee. A player for adaptive MPEG video streaming over the Internet. In *Proceedings 26th Applied Imagery Patter Recognition Workship AIPR-97*, Washington, DC, October 1997. SPIE.

[14] Nicholas Yeadon. *Quality of Service Filters for Multimedia Communications*. PhD thesis, Lancaster University, Lancaster, May 1996.

[15] Wenjun Zeng and Bede Liu. Rate shaping by bock dropping for transmission of MPEG-precoded video over channels of dynamic bandwidth. In *Proceedings of ACM Multimedia'96*, Boston, November 1996.